Claude.Kirchner@loria.fr

Strategic Rewriting

Claude Kirchner

LORIA & INRIA
Nancy, France

Abstract. This is a position paper preparing the round table organized during the 4th International Workshop on Reduction Strategies in Rewriting and Programming. I sketch what I believe to be important challenges of strategic rewriting.

1 What and How

Programming in declarative styles is highly desirable in many domains and allows us to describe *what* we wants to do rather than *how*. Not only it allows for a better understanding of what is programmed, but it permits to make safer programs and systems by allowing to describe their properties and to search or check their proofs.

This is typical of rule based programming (either in its algebraic form or "business" rules one), constraint programming, logic programming, functional programming, spreadsheet programming, ...

Both because the declarative capabilities extend and because we want to express clever controls on them, the description of these controls them self become declarative and therefore them self subject to meta control.

2 What do we program?

Indeed, the first "devices" to be programmed are human: since the very beginning of humanity, we (human) learned for example elementary "recipes" or how to perform elementary arithmetical computations like addition.

Of course computers are today's the big programming construction site. But this is not it, as we would like to program also molecules, cells and complex organism.

In all the cases, even if some implementation details differ, we are faced with the same programming challenges for which the What and How should be specified and possibly executed.

3 Strategies in rule based languages

What makes rule based programming so particularly attractive is first, its ability to discriminate, using pattern matching, the configurations on which we want to act. Second, its capability to perform action specific to this discrimination.

Therefore the understanding of elementary actions is local and usually easy. Of course the global picture, i.e. the semantics of the action of a set of rules is much more elaborated and depends of the control put on the firing of rules.

Because this control could be itself described in a declarative way, it has been called strategies in particular in the term rewriting community. But one speak also of plans or of tactics and tacticals to program the proof search in many theorem provers.

Because programmers tend to model in more abstract ways, the programmed objects become more symbolic and therefore subject to pattern matching. Typical objects are therefore terms, graphs, propositions or clauses, constraints, ...

4 The ELAN experience

When we started the design and implementation of the ELAN language at the beginning of the nineties[15,23], it was first to model inference processes using an almost identity function from the latex file to the ELAN's one, i.e. being able to express the inference process in a syntax very close to the mathematical one. But making such a model executable faced immediately the challenge to describe a strategy for the inference rule application.

For a given rewrite system, we defined a strategy to be just a subset of all the rewrite derivations. This set could be very elaborated and even non recursive. What is interesting here is the design and study of formalisms to describe strategies of interest. Having in mind the design of strategies for ELAN, we formalized this using either proof terms of an appropriate rewriting logic, or as an appropriate rewriting calculus term. The first approach is fully declarative and do not make precise how to describe this set. The former is less declarative but provides us with a constructive way to build the intended derivations.

We called the couple made of a rule system and a strategy a "computational system" but the term "strategic rewriting" inspired by the "strategic programming" terminology, coined by the `Stratego` team, is probably more appealing. The first strategies introduced in the language apart from the standard ones like ";", `identity, repeat, while` ...were `first, dk, dc` (don't know and don't care) followed soon by more elaborated ones like `try` [2].

Semantics The semantics of the ELAN's strategies have been first designed using rewriting logic [15,3]. A more functional semantics [4] leads finally to the emergence, study and application of the rewriting calculus [7,8][1].

Agility As experimented when coding the many applications developed in ELAN, these strategies combinators are quite useful and allow to develop concise models[2]. But they also have limitations in their agility when modeling for example intelligent backtracking when solving CSPs [5], or simply breath first search, or probabilistic choices.

Some flexibility can be achieved using an extended strategy language as developed for ELAN in [1], or by using a reflexive approach [18]. But in general, one would like to have a full strategy language having a clear and simple semantics and good practical efficiency.

5 Strategic challenges

The ELAN experience has been fruitful and inspiring (see `Stratego`[22][3] or the new `Maude` strategy language [20]) and its design and use has provided many feedbacks on what is needed to model strategies.

Here are what I currently see as challenges, of various difficulties, in the research on strategic programming as well as implementation and dissemination.

[1] See also the rewriting calculus web page; `www.loria.fr/~faure/TheRhoCalculusHomePage`.
[2] Examples, applications and contributions are available on the ELAN web page: `elan.loria.fr`
[3] www.stratego-language.org/Stratego/StrategoHistory

Specifying strategies The first challenge is of course to have better ways to specify strategies, taking in mind that we would like to perform of them proofs, evaluation and sharing.

From the existing strategy languages, we should find the most appropriate ways to describe strategies and the best combinators to build them. A typical instance of this problem is to setup a strategy language to conveniently model probabilistic algorithm as well as properties and proof tools for them. A different but related problem consist in describing probabilistic proof methods.

From what has been achieved so far, the rewriting calculus is the framework that I consider currently as the most promising to uniformly express strategies at the appropriate level of detail, in particular because it provides us with an extension of the lambda calculus having matching (modulo) capabilities, explicit non-determinism, and also elaborated exceptions and control mechanisms that are needed to finely express strategies. This still needs a lot of work.

Properties of strategic rewriting As soon as we change the rewriting relation by adding some restrictions on the allowed derivations, we have to re-think the associated proof tools for the generated relation.

The usual properties of confluence, termination, completeness of definition, to mention a few, can no more be checked as before since the relation has been profoundly changed.

So we need to provide new proof tools for strategy guided rewriting. First results have been achieved in this direction since a few year [19,17,12,13], but mainly every previously known result on the rewrite relation generated by a rewrite system should now be generalized to deals with a strategically controlled rewrite system.

Of course the computational capabilities of strategic rewrite systems change with respect to the one of rewrite systems. Therefore, this opens the door to new problems about the decidability and complexity of standard problems like reachability, termination, confluence, ..., for specific strategic rewrite systems. Static analysis, in particular via type systems, model checking or abstract interpretation are promising for guaranteeing suitable soundness program properties.

Strategies evaluations A common belief is that by making the strategies explicit, we gain in clarity and expressiveness, at the risk of a less efficient evaluation.

This is certainly true in a first approximation. But making the control explicit allows us to perform cleaver analysis to simplify or optimize the control and to generate efficient code. For example simplification of strategies are described in [1,13]. The code generation could also benefit of these partial evaluations, and we are still at the beginning of understanding how this could be adapted to strategic rewriting.

Sharing strategies A huge effort is devoted to writing strategies, and this is highly non-trivial. Of course sharing this work and knowledge is of fundamental interest and challenge. A first work in this direction concerns the sharing of strategies between the PVS and Coq proof search programming environments [16].

In general, we wish to disconnect as much as possible the strategies from the data and basic actions to which they apply, building an approach similar to the TOM one, which pattern matching and rewriting abilities are independent of the language on which they are anchored [21][4].

[4] See also `tom.loria.fr`.

Strategic applications and transfers An important effort should be devoted to the in depth transfer of strategy results and technology to practical applications.

Of course a main stream is theorem proving (i.e. proof search specification) and verification as each prover or verifier is centered, either implicitly or explicitly, around clever strategies. With this respect, our experiences in using ELAN or TOM to perform model checking show that strategic programming could be at least as efficient as a dedicated model checker [9,6] but using a much more abstract description of the model.

The application to strategic programming is still in its infancy. Indeed, as shown by the experiences conducted in ELAN, `Maude`, `Stratego` or `Rogue`, the way algorithms are written is quite different when taking a strategic programming point of view. This means that both the researches for to best way to express algorithms using this programming paradigm should be developed, but also that this programming approach should be largely teach, which is not the case currently.

This may change rapidly because of the fundamental interest of the approach due to its strong distinction between the what and how, but also since applications based on XML are becoming extremely popular. This means that symbolic and in particular term structures are omnipresent and that rule base programming becomes a must. Programming languages like XSLT do not take advantage of the explicit use of strategies: strategic programming has here a big chance of exposition and development.

Another field of application is rule based system in the sense of production or business rules [14]. Here one of the fundamental ingredient are strategies, in particular in the rule activation [10]. Since this becomes also a very popular way to understand and model the organization and analysis of companies, markets, ecological, industrial or economical processes, strategic programming will also play a main role here.

As I mention at the beginning, simulating and programming biologic entities is one of the main challenge of this century. Because strategic programming provides both declarative and strategy use, and in particular permits a natural specification of concurrency, it could be one of the best programming paradigm to model complex systems like the biological ones. With this respect the first experiences using `Maude` [11] looks extremely promising and should be developed in the near future from an even more strategy oriented view point.

References

1. P. Borovanský. *Le contrôle de la réécriture: étude et implantation d'un formalisme de stratégies*. Thèse de Doctorat d'Université, Université Henri Poincaré – Nancy 1, France, October 1998. also TR LORIA 98-T-326.
2. P. Borovanský, H. Cirstea, H. Dubois, C. Kirchner, H. Kirchner, P.-E. Moreau, Q.-H. Nguyen, C. Ringeissen, and M. Vittek. ELAN *V 3.6 User Manual*. LORIA, Nancy (France), fifth edition, February 2004.
3. P. Borovansky, C. Kirchner, H. Kirchner, and P.-E. Moreau. ELAN from a rewriting logic point of view. *Theoretical Computer Science*, 2(285):155–185, July 2002.
4. P. Borovanský, C. Kirchner, H. Kirchner, and C. Ringeissen. Rewriting with strategies in ELAN: a functional semantics. *International Journal of Foundations of Computer Science*, 12(1):69–98, February 2001.
5. C. Castro. Building Constraint Satisfaction Problem Solvers Using Rewrite Rules and Strategies. *Fundamenta Informaticae*, 34:263–293, September 1998.
6. H. Cirstea. Specifying Authentication Protocols Using Rewriting and Strategies. In *Third International Symposium on Practical Aspects of Declarative Languages*, volume 1990 of *Lecture Notes in Computer Science*, pages 138–153, Las Vegas, USA, March 2001.
7. H. Cirstea and C. Kirchner. The rewriting calculus — Part I *and* II. *Logic Journal of the Interest Group in Pure and Applied Logics*, 9(3):427–498, May 2001.

8. H. Cirstea, C. Kirchner, L. Liquori, and B. Wack. Rewrite strategies in the rewriting calculus. In B. Gramlich and S. Lucas, editors, *Electronic Notes in Theoretical Computer Science*, volume 86. Elsevier, 2003.
9. H. Cirstea, P.-E. Moreau, and A. Reilles. Rule based programming in java for protocol verification. In N. Martí-Oliet, editor, *Proceedings Fifth International Workshop on Rewriting Logic and its Applications, WRLA 2004, Barcelona, Spain, March 27 – April 4, 2004*, Electronic Notes in Theoretical Computer Science, pages 199–217. Elsevier, 2004.
10. H. Dubois and H. Kirchner. Rule Based Programming with Constraints and Strategies. In K. Apt, A. A. C. Kakas, E. Monfroy, and F. Rossi, editors, *New Trends in Constraints, Papers from the Joint ERCIM/Compulog-Net Workshop, Cyprus, October 25-27, 1999*, volume 1865 of *Lecture Notes in Artificial Intelligence*, pages 274–297. Springer-Verlag, 2000.
11. S. Eker, M. Knapp, K. Laderoute, P. Lincoln, J. Meseguer, and K. Sonmez. Pathway logic: Symbolic analysis of biological signaling. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 400–412, January 2002.
12. O. Fissore, I. Gnaedig, and H. Kirchner. CARIBOO : An induction based proof tool for termination with strategies. In *Proceedings of the Fourth International Conference on Principles and Practice of Declarative Programming*, pages 62–73, Pittsburgh (USA), October 2002. ACM Press.
13. O. Fissore, I. Gnaedig, and H. Kirchner. Simplification and termination of strategies in rule-based languages. In *Proceedings of the Fifth International Conference on Principles and Practice of Declarative Programming (PPDP)*, Uppsala, Sweden, August 2003. ACM Press.
14. C. Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19(1):17–37, September 1982.
15. C. Kirchner, H. Kirchner, and M. Vittek. Designing CLP using Computational Systems. In P. Van Hentenryck and S. Saraswat, editors, *Principles and Practice of Constraint Programming*. The MIT press, 1995.
16. F. Kirchner. Towards a common tactical language : The case of coq and pvs. Master's thesis, Université Denis Diderot, Paris, France, September 2003.
17. H. Kirchner and I. Gnaedig. Termination and normalisation under strategy - Proofs in ELAN. In *Proceedigs of the Third International Workshop on Rewriting Logic and its Applications*, volume 36 of *Elecronic Notes In Theoretical Computer Science*, Kanazawa, JAPAN, September 2000. Elsevier Science Publishers B. V. (North-Holland).
18. H. Kirchner and P.-E. Moreau. A reflective extension of Elan. In J. Meseguer, editor, *Proceedings of the first international workshop on rewriting logic*, volume 4, Asilomar (California), September 1996. Electronic Notes in Theoretical Computer Science.
19. S. Lucas. Termination of context-sensitive rewriting by rewriting. In *Proceedings of the 23rd International Colloquium on Automata, Languages and Programming*, volume 1099 of *Lecture Notes in Computer Science*, pages 122–133. Springer-Verlag, 1996.
20. N. Martí-Oliet, J. Meseguer, and A. Verdejo. Towards a strategy language for Maude. In N. Martí-Oliet, editor, *Proceedings Fifth International Workshop on Rewriting Logic and its Applications, WRLA 2004, Barcelona, Spain, March 27 – April 4, 2004*, Electronic Notes in Theoretical Computer Science, pages 391–414. Elsevier, 2004.
21. P.-E. Moreau, C. Ringeissen, and M. Vittek. A pattern matching compiler for multiple target languages. In G. Hedin, editor, *International Conference on Compiler Construction - CC'2003, Varsovie, Pologne*, volume 2622 of *Lecture notes in Computer Science*, pages 61–76, Apr 2003.
22. E. Visser. Stratego: A language for program transformation based on rewriting strategies. System description of Stratego 0.5. In A. Middeldorp, editor, *Rewriting Techniques and Applications (RTA'01)*, volume 2051 of *Lecture Notes in Computer Science*, pages 357–361. Springer-Verlag, May 2001.
23. M. Vittek. ELAN*: Un cadre logique pour le prototypage de langages de programmation avec contraintes*. Thèse de Doctorat d'Université, Université Henri Poincaré – Nancy 1, October 1994.